In the specification

Please amend paragraph [0022] as follows:

In an embodiment of the present invention, object shell console 102 uses attributes of an interpretive programming language, such as the ~~Java~~ JAVA programming language, to connect to an application that is to be modified without terminating or suspending the application. Once connected, object shell console 102 can extract program data from the application. Program data includes any program information related to the application, including, for example, methods, classes, fields, variable names and values, and arguments passed to or from methods. The program data is displayed to a system operator who uses the program data to modify the executing application without terminating or suspending the executing application. In one embodiment of the present invention, object shell console 102 provides this capability by connecting to the ~~Java~~ JAVA programming language virtual machine (JVM) on which the application to be modified is executing.

Pleas amend paragraph [0023] as follows:

In one embodiment of the present invention, the connection is made using ~~Java~~ JAVA programming language remote method invocation (RMI). ~~Java~~ JAVA programming language RMI is a well-known tool that can be used to access one JVM from another JVM. Use of ~~Java~~ JAVA programming language RMI enables remote invocation and execution of applications and methods. To invoke a remote application (or method), ~~Java~~ JAVA programming language RMI creates a thread to the other application or method. Creation of the new thread using ~~Java~~ JAVA programming language RMI occurs without suspending or terminating the executing application.

Please amend paragraph [0024] as follows:

The application that is connected to and invoked can be either local or remote to administration client 104. Once the connection is established, object shell console 102 has access to the program data of the invoked application. This access results from a feature of the ~~Java~~ JAVA programming language known as introspection. Introspection

was developed for ~~Java~~ JAVA programming language Beans to allow integrated development environments (IDEs) to visually manipulate graphical components to build applications. The object language components that are used for introspection are Class, Method and Field. When object shell console 102 retains a reference to a running object by creating the thread to the application, these components are used to extract the fields and execute the methods of the object class. The fields of the object can be manipulated to create different behaviors in the object. Methods can be re-executed with argument values supplied by a system operator. The values returned by the method's execution can be displayed for analysis.

Please amend paragraph [0029] as follows:

Using the present invention, a system operator can modify an application without terminating or suspending the application, and without waiting for a scheduled maintenance time. Moreover, the update occurs substantially in real-time because the re-execution of a method in an interpretive environment such as ~~Java~~ JAVA programming language repopulates computer memory with the desired contents.

Please amend paragraph [0032] as follows:

The exemplary database class shown in Figure 2 comprises a READ_ORD() method that allows a user to read an order entry from order database 118, a WRITE_ORD() method that allows a user to write and order entry to order database 118, a DELETE_ORD() method that allows a user to delete an order entry from order database 118 and an UPDATE_ORD() method that allows a user to modify an order entry in order database 118. In addition, the database class comprises two INIT() methods. Each INIT() method has a different number of arguments. It would be apparent to those skilled in the art that as ~~Java~~ JAVA programming language allows such overloading of methods to accommodate invocation of methods with different numbers of arguments.

Please amend paragraph [0036] as follows:

The method is then re-executed. By invoking the method while object shell console 102 is connected to the application using ~~Java~~ JAVA programming language

RMI, the memory of the computer on which the application is executing is changed to reflect the new value of the name of the database that was passed in the INIT() method. This update occurs without suspending or terminating the executing application. Because the contents of the memory that the application is using have changed, the executing application continues its execution, using the updated contents of memory. Consequently, the application is modified substantially in real-time without suspension or termination. By modifying the application in this manner, downtime is minimized, thus furthering a primary goal of web-based applications.

Please amend paragraph [0044] as follows:

Figure 6 is a flow diagram for a method for upgrading or otherwise modifying a computer application according to an embodiment of the present invention. In step 602, object shell console 102 attaches to an executing application. As described above, one way to accomplish this attachment is by using ~~Java~~ JAVA programming language RMI to create a thread that executes the application. In step ~~604~~ 601, object console 102 extracts program data from the executing application. In an embodiment of the present invention, step ~~604~~ 601 relies on introspection as described above. This program data includes classes, methods, fields and other data comprising the internal program structure of the application. The program data is displayed to the maintenance person in step 606. Preferably, the program data is displayed in data portion 202 of the GUI of object shell console 102.